VII. Supplementary

A. Advantages of Pseudo-point Analytical Jacobian

The Analytical Jacobian is inefficient in the neural network setting with respect to computational time and space in [14] because of the PointNet based encoder. When large point cloud is fed, the model computes the point feature of every point, then during the feature Jacobian computation, the space of gradient increase linearly with complexity $\mathbb{O}(Nw^2)$ where w is the dimension of inner layers. Thus, PointnetLK-revisited is easily to run out of GPU memory even for ModelNet40 data.

In our model, we compute the Jacobian in an analytical fashion which avoids the numerical unstablities. But differently, the formulation of our feature Jacobian and warp Jacobian is with Pseudo-points. It avoids the linear increase of the computational costs with the input number of points. As our feature gradient in Eq. (15) is on a small number of pseudo-points (size L) and its nearest neighbor in the point cloud, it also avoids taking the large space as PointNetLK-Revisited [14]. If we consider L as same as feature dimension w, the space complexity of the Pseudo-point Analytical Jacobian is $\mathbb{O}(w^2)$, which is much smaller than the $\mathbb{O}(Nw^2)$ given in PointnetLK-revisited[14]. In addition, our implementation is more efficient than $\mathbb{O}(w^2)$ as our $J_{feat} \in \mathbb{O}^{L\times 3\times L}$ only has values on the diagonal in $1, 3$ dimension, thus we reformulated it as $J_{feat}^{'} \in \mathbb{O}^{L\times 3}$ by reducing one dimension. Then the sum-of-products over the 2nd dimension with the warp Jacobian equivalently computes the Jacobian while obtaining $\mathbb{O}(w)$ space and time complexity.

B. Reason for not Comparing with PointNetLK-Revisited on 3DMatch

The Tab. V shows the performance. The registration PointNetLK-Revisited is based on an implementation of a voxelization. However, in the implementation, $voxel\_coords\_p1$ and $voxels\_p1$ are obtained from the aligned point clouds. Then they are transformed with T_{gt} to provide the frames to register. Thus, before registration, their experiment **already knows** the **True Center** and **True Voxel**.

For example, when $voxel = 1$, $voxel_{\text{zero\_mean}} = True$, it subtracts the voxel center before registration, then the algorithm solves the rotation in PointNetLK-Revisited for registration. Until when $voxel_{\text{zero\_mean}} = False$ and $voxel = 1$ it will be a valid experiment. Please refer to the link[4] for a more detailed discussion on this issue.

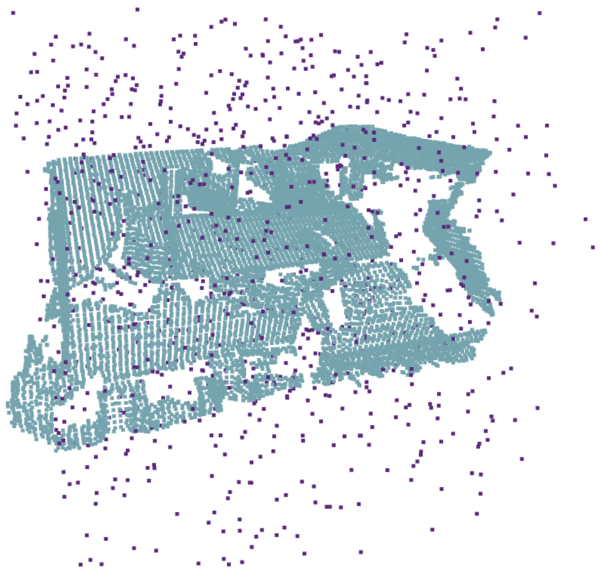
[4]https://github.com/Lilac-Lee/PointNetLK_Revisited/issues/3

	Rot. Error (degrees)		Trans. Error (m)	
Algorithm	RMSE ↓	Median ↓	RMSE ↓	Median ↓
ICP [19]	24.772	**4.501**	1.064	0.149
DCP [20]	53.905	23.659	1.823	0.784
DeepGMR [21]	32.729	16.548	2.112	0.764
PointNetLK [12]	28.894	7.596	1.098	0.260
PointNetLK-Revisited (no voxelization)	**15.996**	4.784	0.738	0.169
PointNetLK-Revisited (voxel=1, zero mean)	14.19	4.82	0.69	0.17
PointNetLK-Revisited (voxel=1, no zero mean)	34.19	5.67	1.754	0.218
PointNetLK-Revisited (8 voxels, 1,000 points)	**7.656**	**2.535**	0.355	**0.096**

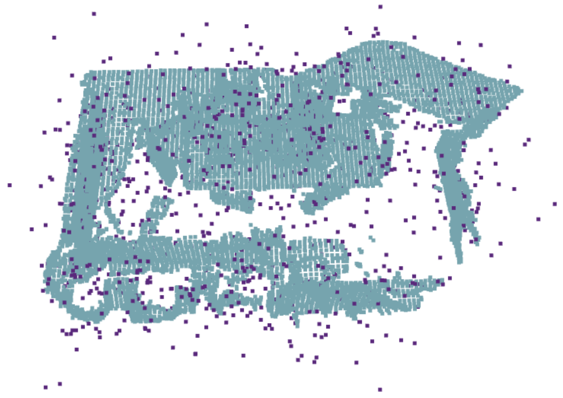
TABLE V: Performance on 3DMatch for PointNetLK-Re with voxelization which is problematic.

C. Pseudo Set Demonstration

In ablation study (Sec. V-J), different pseudo set generation results in different performance. To have a more intuitive understanding, we additionally plot point cloud with pseudo set in Fig. 8. The uniform pseudo set uniformly distributed in space while neighborhood pseudo set are in the neighborhood of surface.



(a) Uniform pseudo set.



(b) Neighborhood pseudo set.

Fig. 8: Demonstration of Pseudo set. Pseudo set is in purple, point cloud is in pine green as in Fig. 1.

D. Algorithm

To better explain our model Sec. IV. We add the Algorithm 1 which is compatible to Fig. 3.

Algorithm 1: Algorithm of our model in Sec. IV and Fig. 3.

```
   Input: $\mathbf{Q}$, $\mathbf{P}$, $\mathbf{S}$, $\varphi$
   Output: $\xi$
1  $\xi = zeros(6)$, $i = 0$
2  Prepare $\mathbf{F_Q} = \varphi(\mathbf{Q}, \mathbf{S})$ and $\varphi(\mathbf{P}, \cdot)$
3  while $i <$ maxiter do
4      $\mathbf{S}^{'} = \mathbf{G}^{-1}(\xi) \cdot \mathbf{S}$
5      $\mathbf{F_P} = \partial\varphi(\mathbf{P}_S, \mathbf{S}^{'})$
6      $\mathbf{J} = \frac{\mathbf{F_P}}{\partial(\mathbf{S}')^T} \frac{\partial(\mathbf{S}^{'})}{\partial\xi^T}$
7      $\mathbf{r} = \mathbf{F_Q} - \mathbf{F_P}$
8      IRLS or directly solve $\Delta\xi$ on $\mathbf{J}\Delta\xi = \mathbf{r}$
9      $\xi = \Delta\xi \oplus \xi$
10     if $max(abs(\Delta\xi)) < threahold$ then
11         break
12     i += 1
```